1. Einleitung

Die Eigennamenerkennung (Named Entity Recognition oder NER) ist eine traditionelle Aufgabe der Computerlinguistik mit dem Ziel der Identifikation und Klassifikation von Eigennamen innerhalb eines Textes. Üblicherweise verwendete Klassen sind Personennamen, Ortsnamen und Organisationen, jedoch werden teilweise wesentlich feinere Differenzierungen vorgenommen.

Eine Unteraufgabe der NER ist das Erkennen von verschachtelten Eigennamen (Nested Named Entity Recognition oder NNER), also Eigennamen der Form *Universität Trier*, wobei die innere Entität *Trier* gesondert von der äußeren Entität *Universität Trier* ausgezeichnet werden soll. Obwohl verschachtelte Eigennamen sprach- und domänenabhängig einen erheblichen Anteil der Gesamtheit ausmachen können, wird dieser Aufgabe in der Forschung nicht immer eine gleichermaßen große Bedeutung zugesprochen.

Historisch wurden verschiedene Methoden angewendet, um diese Aufgaben zu lösen. In der modernen Computerlinguistik werden vielerorts neuronale Netze eingesetzt, da sie in vielen Bereichen der Computerlinguistik zu den besten Ergebnissen führen. Eines der meistverwendeten Sprachmodelle, das auch für (N)NER Anwendung findet, ist BERT (Devlin et al., 2019).

In Li et al. (2019) wurde ein einheitliches Framework auf Basis von BERT entwickelt, das für beide Tasks gleichermaßen geeignet sein soll. Da die Ergebnisse vielversprechend sind, aber noch nicht mit deutschen Korpora getestet wurden, wird dies im Rahmen dieser Arbeit nachgeholt. Das Ziel ist, eine erste Einschätzung zu erhalten, ob sich die Ergebnisse von Li et al. auch auf deutsche Korpora übertragen lassen.

2. „A Unified MRC Framework for Named Entity Recognition“

2.1 Überblick

Li et al. setzen ihren einheitlichen Ansatz um, indem sie die Aufgaben NER und NNER als Ausprägungen der Machine Reading Comprehension (MRC) interpretieren. Die Problemstellung der Extraktion von Personennamen wird also als eine Frage-Antwort-Aufgabe formuliert, bei der als Antwort auf die Frage *„Welche Person wird im Text erwähnt?“* eine oder mehrere Strecken des Eingabetextes mittels Start- und Endeankern markiert werden, die die Antworten enthalten. Analog kann für beliebige weitere Klassen verfahren werden. Als Basis für die Implementierung wird das BERT-Modell (Devlin et al., 2019) verwendet. Da BERT zum Training zwei durch den speziellen [SEP]-Token getrennte Textsequenzen als Eingabe erwartet, können zum Zwecke des (N)NER-Trainings Paare aus (Frage, Suchtext) verwendet werden.

Um die gleichzeitige Anwendung für NER und NNER zu realisieren, werden zwei binäre Klassifikatoren auf den BERT Embeddings trainiert. Diese entscheiden für jedes Token des Suchtextes, ob es der Start bzw. das Ende einer Antwort im Sinne der formulierten Frage ist. Auf diese Weise lässt sich nicht nur einfache NER, sondern auch NNER, sowie die Erkennung möglicherweise überlappender Entitäten umsetzen. Um zuzuordnen, welche Paare aus Anfangs- und Endeanker zusammengehören, wird ein weiterer Klassifikator trainiert, der die dazugehörigen Wahrscheinlichkeiten vorhersagt.

2.2 Datenaufbereitung

Oft liegen Trainingsdaten für NER-Anwendungen im BIO-Format oder im CoNLL-Format vor. Für die Zwecke des MRC Frameworks entschieden sich Li et al. für ein Zwischenformat, das ein Tupel aus (Frage, Antwort, Kontext) abbildet. Aus diesem Zwischenformat haben die Autoren mittels eines Konversionsscripts die Eingabedaten für ihre angepassten BERT-Datasets erstellt.

Bedauerlicherweise wird das besagte Zwischenformat zwar abstrakt beschrieben, die tatsächliche Implementierung jedoch nicht erklärt. So war es erforderlich, aus dem vorliegenden Quellcode[1] die Funktionsweise zu rekonstruieren und ein zusätzliches Konversionsscript zu programmieren, das aus den deutschen Korpora das benötigte Format herstellt (Anhang `scripts/conll2json.py`). Das so erzeugte Datenformat ist dargestellt in Quelltext 2.1. Jeder Satz wird mittels der Schlüssel `"context"` und `"label"` codiert. Dabei enthält `"context"` den Inhalt des vollständigen Satzes als Whitespace-Verkettung aller Tokens der Eingabedatei. `"label"` enthält als Objekt alle vorhandenen Klassen der Named Entities als Schlüssel. Die Werte dieser Schlüssel sind Listen aus `"<start>;<end>"` Strings, die den Indizes der entsprechenden Vorkommnisse innerhalb des `"context"` entsprechen.

Quelltext 2.1: Beispiel aus GermEval2014 für das erforderliche JSON-Format

```
1  {
2    "context": "Aber Borussia Dortmund darf niemals einem
     allein gehören .",
3    "label": {
4      "ORG": [
5        "1;2"
6      ],
7      "LOC": [
8        "2;2"
9      ]
10   }
11 }
```

Aus den Quelldateien der Autoren war nicht ersichtlich, ob und wie dieses Format unter Berücksichtigung von NNER angepasst werden musste. Aus diesem Grund wurde die Annahme getroffen, dass eine Verschachtelung wie in Quelltext 2.1 über die naive Angabe der Indizes bereits ausreichend ist.

[1] https://github.com/ShannonAI/mrc-for-flat-nested-ner/

3. Training für deutsche Korpora

3.1 Deutsche Korpora

Die Wahl der Korpora wurde vorrangig durch Verfügbarkeit und Kompatibilität mit dem gegebenen Framework geleitet. Mindestens ein Korpus sollte NNER abbilden können, weshalb die Entscheidung für GermEval2014 (Benikova et al., 2014) getroffen wurde. Mit MultiNERD (Tedeschi & Navigli, 2022) wurde ein Korpus betrachtet, das besonders durch die Vielzahl an ausgezeichneten Entityklassen interessant ist. Zusätzlich wurde WikiANN (Pan et al., 2017) aufgenommen, allerdings hauptsächlich aus opportunistischen Gründen bezüglich Verfügbarkeit und Aufwand. Eine Übersicht der verwendeten Korpora ist in Tabelle 3.1 dargestellt.

Mit Europeana Newspapers (Neudecker, 2016) wurde ein weiteres Korpus in Betracht gezogen, jedoch konnte dieses nicht verwendet werden, da hier keine Trennung der Sätze stattfindet. Der Versuch, mittels Anwendung eines Satztokenizers die Trennung automatisiert vorzunehmen, lieferte keine brauchbaren Ergebnisse. Der Hauptgrund dafür ist vermutlich, dass die Datenquelle dieses Korpus eine OCR-basierte Digitalisierung von Zeitungsartikeln ist, die keine ausreichende Qualität bietet. Die Problematik der Satztrennung wird von Neudecker ebenfalls beklagt.

Tabelle 3.1: Verwendete deutsche Korpora

Korpus	**Sätze**	**Tokens**	**Nested**	**Entityklassen**
GermEval2014	31,000	>590,000	Ja	4
WikiANN	40,000	>390,000	Nein	3
MultiNERD[1]	156,800	2,700,000	Nein	15

3.2 Deutsches BERT-Modell

Während meine früheren Arbeiten auf `bert-base-german-cased` (Chan et al., 2019) basierten, fiel in dieser Arbeit die Entscheidung für das modernere `gbert-base` (Chan et al., 2020), das mit einer deutlich besseren Performance beworben wird.

[1]Die hier gelistete Anzahl der Sätze und Tokens entspricht der Gesamtgröße des Korpus. Zur Verwendung in dieser Arbeit siehe Kapitel 3.

Letztendlich wurde diese Entscheidung aus rein pragmatischen Gründen getroffen, da bereits zu Beginn absehbar war, dass in dieser Arbeit lediglich ein Modell getestet werden kann. Gleichzeitig ist die Wahl des Basismodells nur eine der vielen Stellschrauben, die Einfluss auf die Performance des Frameworks haben. Einige dieser Stellschrauben werden im Folgenden an gegebener Stelle aufgezeigt, sowie in Kapitel 5 überblicksartig zusammengefasst.

3.3 Debugging

Im Repository der Autoren befinden sich Scripts, die eine Reproduktion der Ergebnisse aus Li et al. (2019) ermöglichen sollen. Als Basis für die Anpassung an die deutschen Korpora wurde `scripts/mrc_ner/reproduce/genia.sh` verwendet. Bevor eine angepasste Version dieses Scripts verwendet werden konnte, mussten jedoch zunächst diverse Versionskonflikte der installierten Python-Pakete behoben werden. Dieses Debugging war wie üblich sehr zeitaufwändig, da die produzierten Fehlermeldungen nicht immer einen offensichtlichen Hinweis auf das zugrundeliegende Problem lieferten. Die relevanten finalen Versionen der verwendeten Pakete sind im Anhang `versions.txt` aufgeführt. Überraschenderweise zeigten sich große Diskrepanzen zwischen den auf diese Weise ermittelten Versionen und denen, die in der mitgelieferten `requirements.txt` gelistet sind.

Nachdem Konflikte in der Python-Installation bereinigt waren, wurden die in Abschnitt 2.2 erläuterten Probleme der Datenformate offenbart. Im Zuge dessen musste ein Konversionsscript entwickelt werden (Anhang `scripts/conll2json.py`). Aus den so erhaltenen JSON-Daten wurde mit `ner2mrc/genia2mrc.py` anschließend das benötigte MRC Format generiert, das letztendlich die Eingabe für das Training darstellt.

Ein weiterer Faktor, der in Kapitel 2 bereits angedeutet wurde, sind die erforderlichen Formulierungen der Fragen bzw. Queries, anhand derer das Modell Antworten finden soll. Während die einfachste Methode darin besteht, ein einzelnes Wort als Query zu verwenden, kann ebenfalls kurze Umschreibung oder gar eine voll ausformulierte Frage zur Entityklasse genutzt werden. Weitere Untersuchungen zu diesem Aspekt finden sich in Li et al. (2019).

Ein erster Versuch, eine mitgelieferte Query-Definition der Autoren unverändert zu übernehmen, ließ das Training zwar erfolgreich abschließen, in der Evaluation wurde jedoch ein enttäuschender F1-Score von ca. 0.51 erreicht. Dies ist auf mangelndes Verständnis des Frameworks zurückzuführen, da beim zweiten Blick klar wurde, dass die unreflektiert übernommenen Queries auf Chinesisch formuliert und somit für ein deutsches Modell nicht nutzbar waren.

Ein weiteres Problem zeigte sich beim Versuch, das Modell für MultiNERD zu trainieren. Aufgrund der schieren Größe der Datenmenge (s. Tabelle 3.1) war es nicht machbar, eine Konfiguration der Hyperparameter zu finden, die ein Training auf einer GPU in vertretbarer Zeit ermöglichte. Da die mitgelieferten Reproduktionsscripts ebenfalls Konfigurationen für ein Multi-GPU Setup enthielten, wurde zunächst versucht, diese entsprechend zu übernehmen. Es zeigten sich aber zu große Unterschiede zwischen den Serverarchitekturen der Autoren und dem mir zur Verfügung stehenden Server. Daher wurde der Versuch aufgegeben, das gesamte MultiNERD Korpus zu verwenden. Stattdessen wurde die Größe des Korpus auf $\frac{1}{6}$ (ca. 26133 Sätze) reduziert, was ein Training auf einer GPU ermöglichte.

Alle finalen Trainings- und Evaluationsscripts können im Anhang eingesehen werden (Anhang `train/` und Anhang `evaluate/`).

3.4 Anpassung der Queries

Da GermEval2014 der erste für dieses Experiment verwendete Datensatz war, wurden zunächst Ein-Wort-Queries als Platzhalter eingefügt, um erste plausible Ergebnisse zu erhalten. Es schien für den Fortschritt des Experimentes wichtig, nun weitere Korpora zu untersuchen. Deshalb wurde an dieser Stelle auf ein anschließendes Training mit angepassten Queries für GermEval2014 verzichtet. Weil MultiNERD eine große Zahl an Entityklassen enthält und in Tedeschi und Navigli (2022) Definitionen bereitgestellt werden, wurden diese ins Deutsche übersetzt und als Queries verwendet. Für das anschließende Training mit WikiANN wurden ebenfalls die übersetzten Definitionen verwendet.

Nach dem erfolgten Training für alle drei Korpora stellte sich die Frage, ob ein erneuter Trainingsdurchlauf mit angepassten Queries für GermEval2014 in Betracht gezogen werden sollte. Die Erfahrung zeigte nun, dass die längeren Queries starken Einfluss auf den Ressourcenbedarf des Trainings haben: Das Training mit vollständigen Definitionen erfordert ca. 50% mehr Zeit und VRAM als das Training mit Ein-Wort-Queries. In Anbetracht der ohnehin erheblichen Trainingsdauer (vgl. Tabelle 4.1) wurde daher im Rahmen dieser Arbeit kein weiteres Training mit angepassten Queries durchgeführt.

Die finalen Queries können im Anhang eingesehen werden (Anhang `queries/`).

4. Evaluation

4.1 Übersicht der Trainingsergebnisse

Es folgen in Tabelle 4.1 die Ergebnisse der Evaluation des MRC Frameworks. Insgesamt sind die Ergebnisse vergleichbar mit denen englischsprachiger Korpora von Li et al. Speziell die mit MultiNERD erreichten Scores übertreffen diese teilweise sogar deutlich. Auch wenn die Performance auf GermEval2014 ungefähr 6%–8% schlechter abschneidet, sind die Ergebnisse wesentlich besser als die ursprünglichen Systeme der GermEval Konferenz (Benikova et al., 2014). Außerdem sind sie vergleichbar mit moderneren Modellen wie Riedl und Padó (2018), die auf diesem Korpus trainiert wurden. Für das deutsche WikiANN gibt es kaum Vergleichswerte, jedoch wird in Schiesser (2023) ein etwas besserer F1-Score von 0.8892 berichtet.

Die Evaluation fand mit angepassten Skripten der Autoren statt (Anhang `evaluate/`). Metriken werden gemäß Li et al. (2019) als span-level micro-average berichtet.

Tabelle 4.1: Evaluation des MRC Frameworks

Korpus	**dev**			**test**			**Trainingsdauer**
	f1	**precision**	**recall**	**f1**	**precision**	**recall**	**(in Min.)**
GermEval2014	0.8255	0.8379	0.8136	0.8111	0.8369	0.7867	200
WikiANN	0.8578	0.8625	0.8532	0.8604	0.8545	0.8665	223
MultiNERD	0.8823	0.8986	0.8665	0.8841	0.8994	0.8692	644

4.2 Interpretation der Trainingsergebnisse

Es gibt verschiedene Erklärungsansätze für die Beobachtung, dass GermEval2014 deutlich schlechter abschneidet als die anderen Korpora. Zunächst ist es der einzige Datensatz des NNER-Tasks. Wegen der höheren Komplexität dieser Aufgabe sind die Ergebnisse grundsätzlich als Sonderfall zu betrachten. Zudem ist GermEval2014 auch das kleinste Korpus (vgl. Tabelle 3.1), wodurch möglicherweise kein ausreichendes Training möglich war. Darüber hinaus war es, wie in Abschnitt 3.4 beschrieben, das erste Korpus, mit dem trainiert wurde, und hatte lediglich Ein-Wort-Queries als Basis. Es ist mit Sicherheit davon auszugehen, dass dieser Faktor die Performance entscheidend reduziert hat.

WikiANN und MultiNERD sind maschinell generierte Silver-Standard-Korpora. Das führt zu einer großen Menge an Trainingsdaten, die jedoch nicht immer von hoher Qualität sind. Trotz der guten Evaluationsergebnisse muss also angenommen werden, dass auch die trainierten Modelle einen gewissen Fehleranteil aufweisen, der rein numerisch nicht sichtbar ist.

Infolgedessen sind die Ergebnisse in dieser Form nur eingeschränkt vergleichbar. Um ein tatsächlich faires Bild der Performance des MRC Frameworks zu erhalten, sollten weitere Experimente durchgeführt werden. Mögliche Faktoren, die dabei berücksichtigt werden könnten, werden in Kapitel 5 aufgeführt.

5. Ausblick

Trotz der aufschlussreichen Ergebnisse bezüglich der Performance des MRC Frameworks auf deutschen Korpora gibt es an vielen Stellen Bedarf für weitere Untersuchungen. Im Rahmen dieser Arbeit war vor allem angesichts der hohen Trainingsdauer nicht möglich, auf alle denkbaren Konfigurationen einzugehen. Im Folgenden werden einige Aspekte aufgeführt werden, von denen zu erwarten ist, dass sie großen Einfluss auf die Performance des Frameworks haben.

- **Hyperparameter:** Die Anzahl der Epochen, die Batch-Size und die Lernrate sind offensichtliche Hyperparameter, mit denen experimentiert werden könnte. Mögliche Versuche sind allerdings direkt gebunden an die verfügbaren Ressourcen.
- **Queries:** Wie in Abschnitt 3.4 beschrieben, ist die Formulierung der Queries ein entscheidender Faktor, der von Li et al. genauer beleuchtet wurde. Allerdings wurde nicht im Speziellen die deutsche Sprache diskutiert. Obwohl zu vermuten ist, dass sich die Beobachtungen der Autoren auch auf das Deutsche übertragen lassen, bleibt dies zu überprüfen.
- **Korpora:** Da die in dieser Arbeit gewählten Korpora lediglich eine kleine Auswahl der verfügbaren Daten darstellen, ist es naheliegend, die Untersuchungen auf weitere Korpora auszubreiten, sowie eine technische Konfiguration zu erreichen, in der ein Training mit dem vollständigen MultiNERD Korpus möglich ist.
- **BERT-Modell:** Das verwendete Modell `gbert-base` wirkt zwar vielversprechend, trotzdem ist es möglich, dass andere Modelle, die bspw. auf $\text{BERT}_{\text{LARGE}}$ oder RoBERTa basieren, bessere Ergebnisse liefern. Eine andere Modellarchitektur, die überhaupt nicht auf BERT basiert, ist zwar theoretisch denkbar. Da jedoch das gesamte MRC Framework auf die BERT-Architektur ausgelegt ist, wäre dies nicht ohne erheblichen Aufwand umsetzbar.

6. Fazit

In dieser Arbeit wurde das von Li et al. (2019) entwickelte einheitliche Framework für NER und NNER erstmalig auf deutschen Korpora angewendet und evaluiert. Obwohl Li et al. sämtlichen Quellcode offenlegen, war die Anwendung mit großem Aufwand verbunden, was einerseits auf die rasche Weiterentwicklung und begrenzte Abwärtskompatibiltät im Bereich des Machine Learning mit Python zurückzuführen ist. Andererseits waren entscheidende Details der erforderlichen Datenformate bedauerlicherweise nicht nachvollziehbar in Li et al. (2019) oder dem dazugehörigen Quellcode dokumentiert. Die anschließende Evaluation hat gezeigt, dass die Performance auf den deutschen Korpora vergleichbar mit den von Li et al. erhaltenen Ergebnissen ist. Gleichzeitig konnten aufgrund der Komplexität einer solchen Untersuchung nicht alle Facetten beleuchtet werden, die für eine allgemeingültige Aussage über die Performance des Frameworks, sowie den potenziellen Mehrwert im Bereich der (N)NER notwendig wären.

A. Anhang

Die folgenden Dateien sind in der E-Mail angehängt und befinden sich zusätzlich im Repository unter: https://github.com/thielenf/nlp-ha/tree/main/appendix

```
appendix/
├── evaluate/
│   ├── germeval_evaluate.sh
│   ├── multinerd_evaluate.sh
│   └── wikiann_evaluate.sh
├── queries/
│   ├── germeval.json
│   ├── multinerd.json
│   └── wikiann.json
├── scripts/
│   └── conll2json.py
├── train/
│   ├── germeval_train.sh
│   ├── multinerd_train.sh
│   └── wikiann_train.sh
└── versions.txt
```

Literatur

Benikova, D., Biemann, C., & Reznicek, M. (2014). NoSta-D Named Entity Annotation for German: Guidelines and Dataset. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 2524–2531. http://www.lrec-conf.org/proceedings/lrec2014/pdf/276_Paper.pdf

Chan, B., Pietsch, M., Möller, T., & Soni, T. (2019). bert-base-german-cased · Hugging Face. https://huggingface.co/bert-base-german-cased

Chan, B., Schweter, S., & Möller, T. (2020). German's Next Language Model. *Proceedings of the 28th International Conference on Computational Linguistics*, 6788–6796. https://doi.org/10.18653/v1/2020.coling-main.598

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Li, X., Feng, J., Meng, Y., Han, Q., Wu, F., & Li, J. (2019). A Unified MRC Framework for Named Entity Recognition. *arXiv preprint arXiv:1910.11476*.

Neudecker, C. (2016). An Open Corpus for Named Entity Recognition in Historic Newspapers. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 4348–4352. https://aclanthology.org/L16-1689

Pan, X., Zhang, B., May, J., Nothman, J., Knight, K., & Ji, H. (2017). Cross-lingual Name Tagging and Linking for 282 Languages. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1946–1958. https://doi.org/10.18653/v1/P17-1178

Riedl, M., & Padó, S. (2018). A Named Entity Recognition Shootout for German. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 120–125. https://doi.org/10.18653/v1/P18-2020

Schiesser, M. (2023). mschiesser/ner-bert-german · hugging face. https://huggingface.co/mschiesser/ner-bert-german

Tedeschi, S., & Navigli, R. (2022). MultiNERD: A Multilingual, Multi-Genre and Fine-Grained Dataset for Named Entity Recognition (and Disambiguation).

Findings of the Association for Computational Linguistics: NAACL 2022, 801–812. https://doi.org/10.18653/v1/2022.findings-naacl.60